

|  |  |
| --- | --- |
| **MANCEAU Charlène** | **Branche :**  Informatique et Systèmes d’Information |
| **LEYRIS Théolise** | **Semestre**: Automne 2018 |

**A travers l’UE IF26, nous avons conçu une application mobile en Android et IOS que nous avons nommé *Pills*. Il s’agit d’une application pour les rappels de prise de médicaments.**

**Conception d’une application mobile en Android et IOS**

|  |
| --- |
| **UE :** IF26 |
| **Responsable :** Marc LEMERCIER |
|  |

**Pills**

**Application de rappels de Médicaments**

Sommaire

[Sommaire 2](#_Toc535244127)

[1. Introduction 3](#_Toc535244128)

[2. Analyse 4](#_Toc535244129)

[2.1. Conception de la base de données 4](#_Toc535244130)

[2.1.1. Table Médicament 4](#_Toc535244131)

[2.1.2. Table Rappel 4](#_Toc535244132)

[2.2. Conception globale de l’application 5](#_Toc535244133)

[2.2.1. Aujourd’hui 5](#_Toc535244134)

[2.2.2. Mes Médicaments 6](#_Toc535244135)

[2.2.3. Mes rappels 9](#_Toc535244136)

[2.2.4. Données Personnelles 10](#_Toc535244137)

[2.3. Conception de l’application Android 11](#_Toc535244138)

[2.4. Conception de l’application IOS 11](#_Toc535244139)

[3. Réflexions et démarches de sécurisation 13](#_Toc535244140)

[3.1. Intégration des directives RGPD dans notre application 13](#_Toc535244141)

[3.2. La sécurité des applications mobiles 14](#_Toc535244142)

[4. Conclusion 15](#_Toc535244143)

# Introduction

Dans le cadre du projet de IF26, nous avions à réaliser une même application mobile dans les deux environnements Android et IOS. Pour notre choix d’application, nous avons eu l’idée de développer une application pour les rappels de prise de médicaments, que nous avons nommée ***Pills*.** Nous avons regardé sur Google Play les applications sur le même thème, et celle qui se rapproche le plus de notre idée de départ est l’application *MyTherapy*.

L’objectif de notre application est de simplifier la prise de médicaments par le biais de rappels, et présente idéalement les fonctionnalités suivantes. Premièrement, l’utilisateur pourra créer un médicament et l’ajouter à sa liste. Pour cela, il devra renseigner quelques informations sur le médicament tel que son nom ou son fabricant. Par la suite, l’utilisateur pourra à tout moment modifier les informations d’un médicament, ou bien le supprimer complètement. Une fois un médicament enregistré, l’utilisateur aura la possibilité de créer des rappels liés à celui-ci. Il renseignera alors certaines caractéristiques du rappel tel que l’heure ou la répétition. L’utilisateur pourra ajouter autant de rappels qu'il le souhaite pour un même médicament, et, comme pour les médicaments, il aura la possibilité de les modifier ultérieurement ainsi que de les supprimer. Par ailleurs, lorsque l’utilisateur supprimera un médicament, cela supprimera tous les rappels liés. De plus, l’application affichera sur la page d’accueil les rappels du jour, et enverra une notification à l’heure du rappel. L’utilisateur pourra alors valider la prise du médicament ou l’ignorer. Une autre fonctionnalité un peu annexe qui peut être intéressante de développer par la suite, est d’ajouter un rappel/une notification en fin de stock du médicament pour penser à en racheter. Ces fonctionnalités sont ce que nous aurions idéalement aimé réaliser, cependant nous n’avons pas développer l’ensemble de ces fonctions. En effet, nous n’avons pas eu le temps de gérer les notifications, et la fonctionnalité secondaire de rappel en fin de stock du médicament.

Par ailleurs, nous avons choisi cette application car nous pensons que le thème de la santé/bien-être est plutôt à la mode au niveau des applications, et qu’il était donc intéressant de s’y pencher. Avec cette application, nous pouvions améliorer nos compétences, puisqu’elle est plutôt complète avec de nombreuses fonctionnalités sans être trop complexe. La principale difficulté est la gestion des dates des rappels. De plus, cette application nous permettait de gérer de la persistance locale de données avec notamment les fonctionnalités de création, modification et suppression des médicaments et des rappels. En effet, pour que l’application soit intéressante, il faut qu’elle stocke la liste des médicaments et des rappels, et donc qu’elle persiste. L’application nous permettait également de développer des interfaces utilisateurs variées tel que des formulaires pour les fonctionnalités d’ajout/modification des éléments, des listes dynamiques d’affichage, des boutons, un menu… Ainsi, l’application que nous avons choisie, nous permettait de valider les compétences demandées.

# Analyse

## Conception de la base de données

La base de données de notre application se compose de deux tables permettant de stocker les éléments principaux qui sont les médicaments et les rappels. Ces deux tables sont décrites en détail ci-après.

### Table Médicament

La première table que nous avons créée est la table **Medicament** pour la persistance des objets médicaments. Elle se compose des attributs suivants :

* Un attribut *id* qui s’auto-incrémente permettant d’identifier de manière unique un médicament. Il s’agit de la clé primaire de la table.
* Un attribut *nom* qui est simplement le nom du médicament que l’utilisateur renseigne lors de l’ajout. Cet attribut ne peut pas être nul car il sert à distinguer les différents médicaments sur l’interface pour l’utilisateur.
* Un attribut *type* qui est le type de médicament (comprimé, granules, sirop…) renseigné par l’utilisateur. Cet attribut est optionnel.
* Un attribut *fabricant* qui correspond au fabricant du médicament renseigné par l’utilisateur. Cet attribut est optionnel lors de l’ajout d’un médicament.
* Un attribut *stock* qui correspond au stock de médicaments de la boîte, du flacon… Cet attribut est optionnel, et serait utile pour la fonctionnalité de rappel en fin de stock.

Création de la table Medicament :

**"CREATE TABLE IF NOT EXISTS "** + ***TABLE\_MEDICAMENT*** + **"("** +

***ATTRIBUT\_ID\_MEDICAMENT*** + **" INTEGER primary key AUTOINCREMENT,"** +

***ATTRIBUT\_NOM*** + **" TEXT not null, "** +

***ATTRIBUT\_FABRICANT*** + **" TEXT, "** +

***ATTRIBUT\_TYPE*** + **" TEXT, "** +

***ATTRIBUT\_STOCK*** + **" REAL"** +

**")"**

### Table Rappel

Après avoir créée la table Médicament, nous avons réalisé notre deuxième table, la table **Rappel**, pour la persistance des objets rappels. Elle se compose des attributs suivants :

* Un attribut *id\_rappel* qui s’auto-incrémente permettant d’identifier de manière unique un rappel. Il s’agit de la clé primaire de la table.
* Un attribut *id\_med* qui est l’identifiant du médicament auquel le rappel est lié. Il s’agit d’une clé étrangère qui référence l’attribut id de la table Médicament.
* Un attribut *heure* qui est l’heure de la journée (sous la forme d’une chaîne de caractères : HH:MM) à laquelle le rappel est défini.
* Un attribut *repetition* qui est la fréquence à laquelle le rappel se répète : tous les jours, tous les 2 jours… Pour simplifier, nous avons considéré qu’un rappel pouvait uniquement être répété tous les X jours. X est donc défini par l’attribut repetition. Par défaut, la répétition est égale à 1, c’est à dire que le rappel est quotidien, il est répété tous les jours.
* Un attribut *statut* qui permet de définir si le rappel a été pris ou non. Par défaut, le statut est égal à 0 (false), et il passe à 1 (true) lorsque le rappel a été pris. Le statut doit être réinitialisé tous les jours.
* Un attribut *prochain\_rappel* qui définit la date du prochain rappel (sous la forme d’une chaine de caractères : DD/MM/YYYY). Cet attribut est comparé à la date courante pour savoir si le rappel doit être pris aujourd’hui.

Création de la table Rappel

**"CREATE TABLE IF NOT EXISTS " + *TABLE\_RAPPEL* + "(" +**

***ATTRIBUT\_ID\_RAPPEL* + " INTEGER primary key AUTOINCREMENT," +**

***ATTRIBUT\_ID\_MED* + " INTEGER not null, " +**

***ATTRIBUT\_HEURE* + " TEXT, " + *//Heure au format HH:MM***

***ATTRIBUT\_REPETITION* + " INTEGER DEFAULT 1, " +**

***ATTRIBUT\_STATUT* + " INTEGER DEFAULT 0, " + *//Booléen : 0 pour false et 1 pour true***

***ATTRIBUT\_PROCHAIN\_RAPPEL* + " TEXT, " + *//Date au format "DD/MM/YYYY"***

**" FOREIGN KEY(" + *ATTRIBUT\_ID\_MED* + ") REFERENCES " + *TABLE\_MEDICAMENT* + "(" + *ATTRIBUT\_ID\_MEDICAMENT* + ") " +**

**")"**

Pour la version IOS de l’application, la table Rappel est un peu différente. L’attribut *heure* est un objet Date. C’est cet attribut qui définit la date du prochain rappel et qui sera comparé à la date courante pour savoir si le rappel est aujourd’hui. L’attribut *prochain\_rappel* est remplacé par l’attribut *dernier\_rappel*, un objet Date également. Cet attribut correspond tout simplement à la date du *dernier\_rappel*. Il n’a pas d’utilité particulière, si ce n’est fournir une sorte d’historique des rappels.

## Conception globale de l’application

Au niveau de la conception globale de notre application, chaque page se compose d’une toolbar en haut avec une icône de menu burger et le nom de l’application, et d’une zone de contenu. Le menu est un panneau latéral et se constitue d’un en-tête avec le nom de l’application, et des items suivants : Aujourd’hui, Mes médicaments, Mes rappels et Données Personnelles.

### Aujourd’hui

Lorsque l’utilisateur sélectionne l’item “Aujourd’hui” du menu, il est amené sur une page présentant la liste des rappels qu’il doit prendre à la date d’aujourd’hui. En cliquant sur un des rappels, l’utilisateur est envoyé sur une nouvelle page récapitulant les informations du rappel et lui donnant la possibilité de valider la prise du rappel.

La page Aujourd’hui est également la page d’accueil lorsqu’on lance l’application.

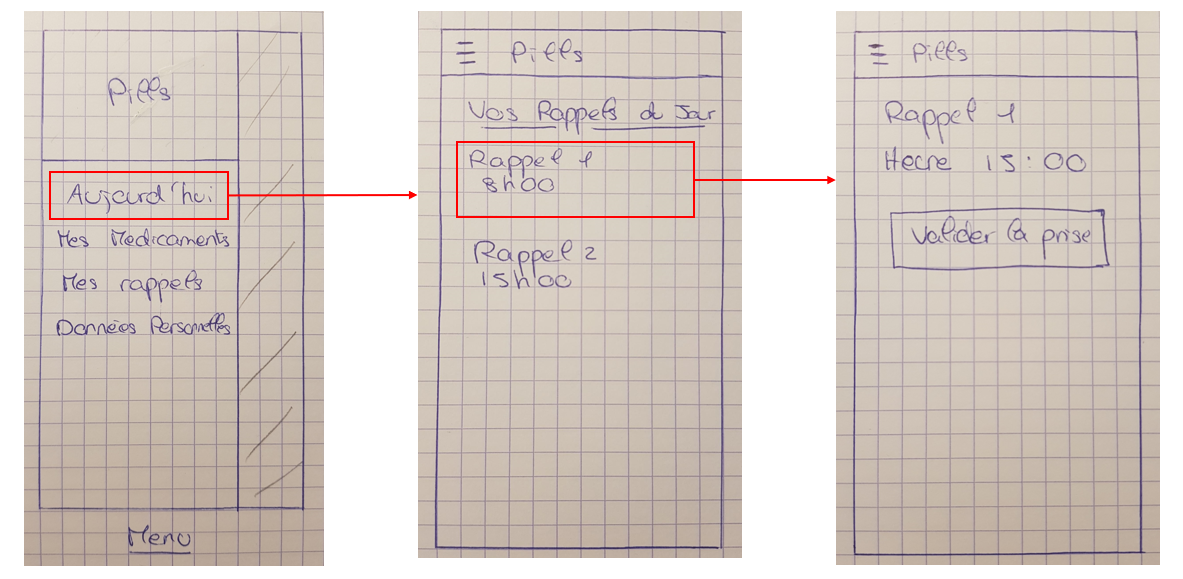


Figure : Maquettes du menu, de la page Aujourd'hui et de la validation de la prise d'un rappel

### Mes Médicaments

L’utilisateur peut consulter la liste des médicaments qu’il a enregistrés via l’item “Mes médicaments” du menu. La page se constitue donc de la liste des médicaments, mais également d’un bouton Ajouter.

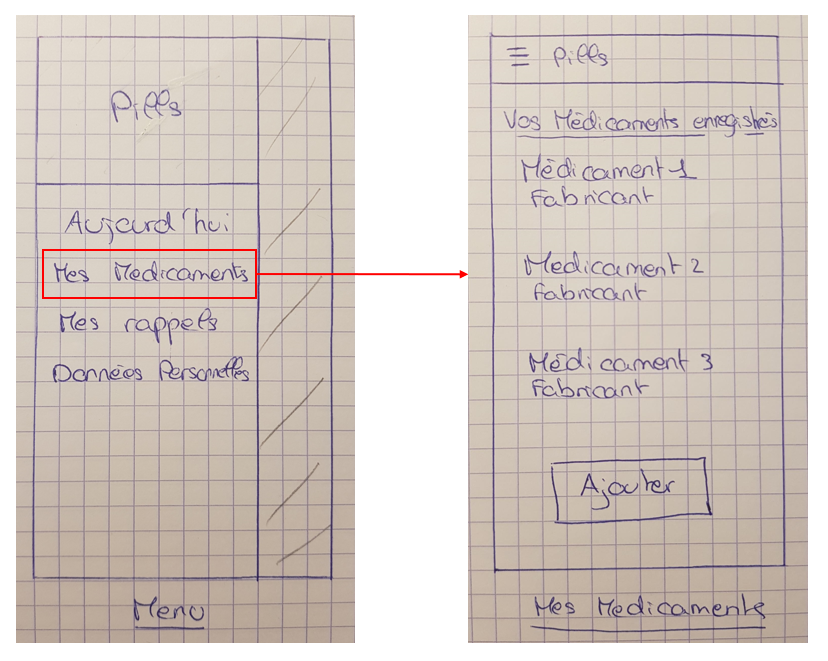


Figure : Maquette de la page Mes médicaments

L’utilisateur peut alors réaliser différentes fonctionnalités. Il peut tout d’abord ajouter un nouveau médicament à partir de cette page grâce au bouton Ajouter. En cliquant dessus, il sera dirigé vers un formulaire où il devra renseigner les caractéristiques du médicaments (nom, type de prise, fabricant, quantité dans la boîte). Seul l’attribut nom est obligatoire pour la validation de l’ajout du médicament. Une fois validé, le médicament est visible dans la liste des médicaments.

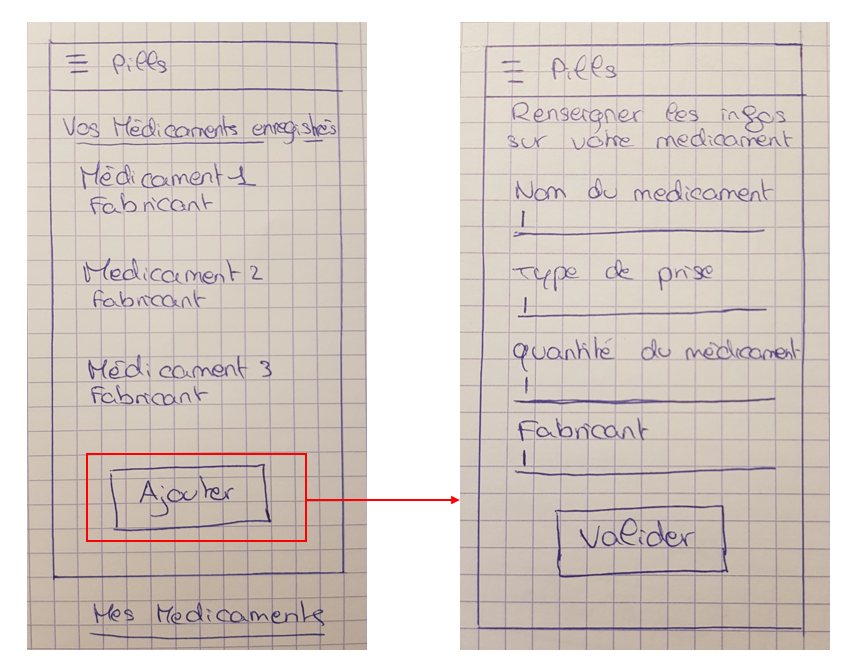


Figure : Maquettes de l'ajout d'un médicament

La deuxième action que l’utilisateur peut faire sur la page “Mes Médicaments” est de consulter les informations d’un médicament en cliquant sur celui-ci dans la liste. Il est alors dirigé vers une nouvelle page où il trouve l’ensemble des caractéristiques du médicament ainsi que les rappels associés. L’utilisateur a également la possibilité d’ajouter un rappel, de consulter les informations d’un rappel, de modifier le médicament et de supprimer le médicament.

Le bouton Supprimer supprimera le médicament et les rappels associés. Le bouton Modifier enverra l’utilisateur sur le même formulaire que lors de l’ajout du médicament mais avec les champs pré-remplis. Lorsqu’il valide, le médicament est alors modifié.

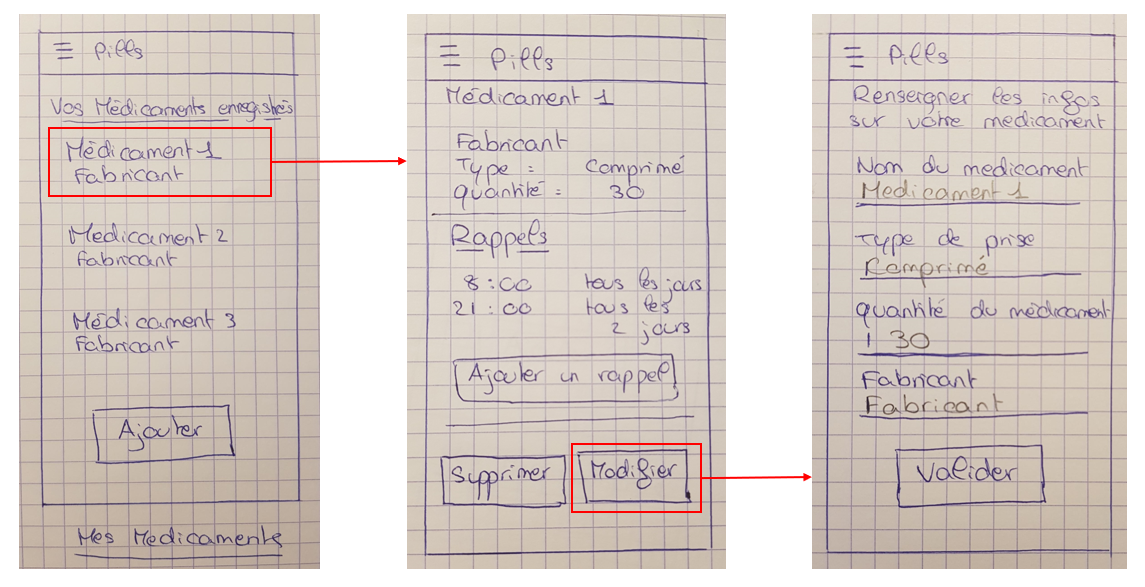


Figure : Maquettes de la page d'un médicament, et de sa modification

Comme dit précédemment, sur la page d’un médicament, l’utilisateur peut aussi ajouter un rappel via le bouton correspondant. L’utilisateur remplit alors un formulaire en renseignant les informations du rappel : son heure et sa répétition (c’est à dire si l’utilisateur veut que le médicament soit répété tous les jours, tous les 2 jours, tous les X jours…). Seul l’horaire du rappel est obligatoire à renseigner, et par défaut la répétition est fixée à tous les jours. Le rappel est automatiquement lié au médicament concerné. Une fois validé, le rappel est ajouté et visible sur la page du médicament.



Figure : Maquettes de l'ajout d'un rappel

Sur la page du médicament, l’utilisateur peut alors cliquer sur le rappel et accéder à une nouvelle page avec les caractéristiques de celui-ci, et a également la possibilité de le modifier ou de le supprimer. Ces fonctionnalités fonctionnent de la même façon que pour les médicaments.

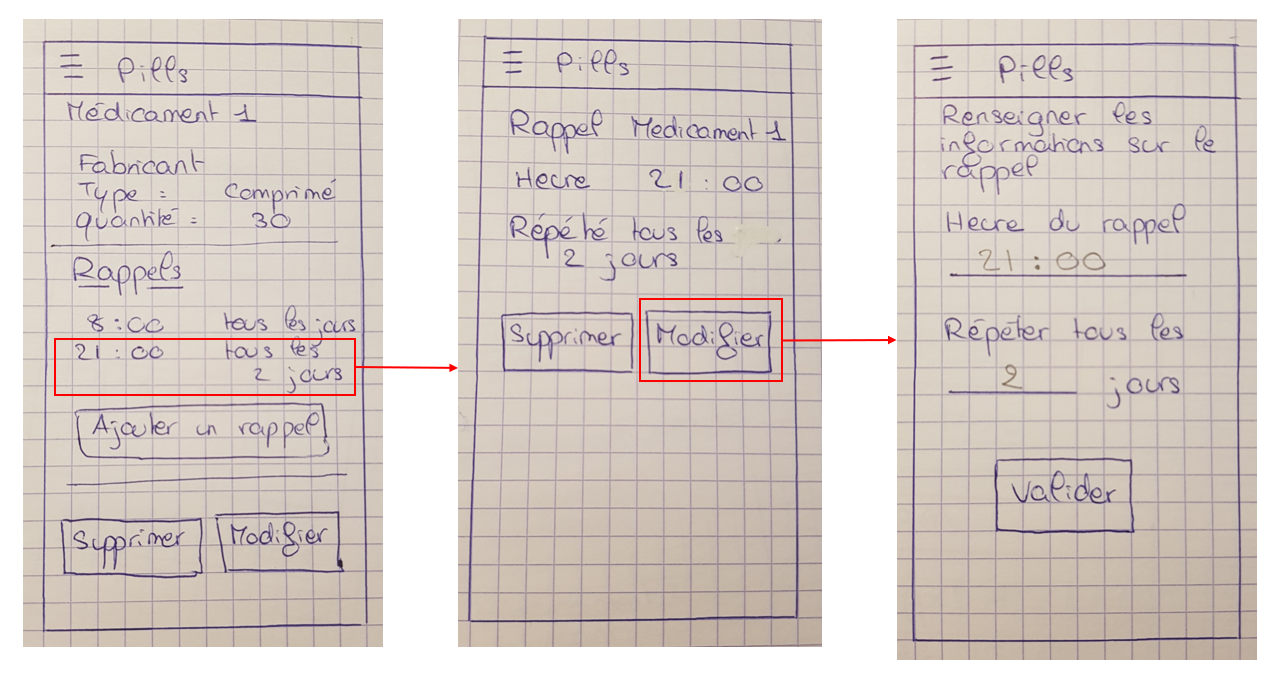


Figure : Maquettes d'un rappel, et de sa modification

### Mes rappels

Via l’item “Mes rappels” du menu, l’utilisateur a accès à la liste de tous les rappels créés, peu importe les médicaments concernés. Il peut consulter les informations d’un rappel spécifique en cliquant sur celui-ci, et il sera dirigé vers la même page que précédemment, composée des caractéristiques du rappel, et des fonctions de modification et de suppression.

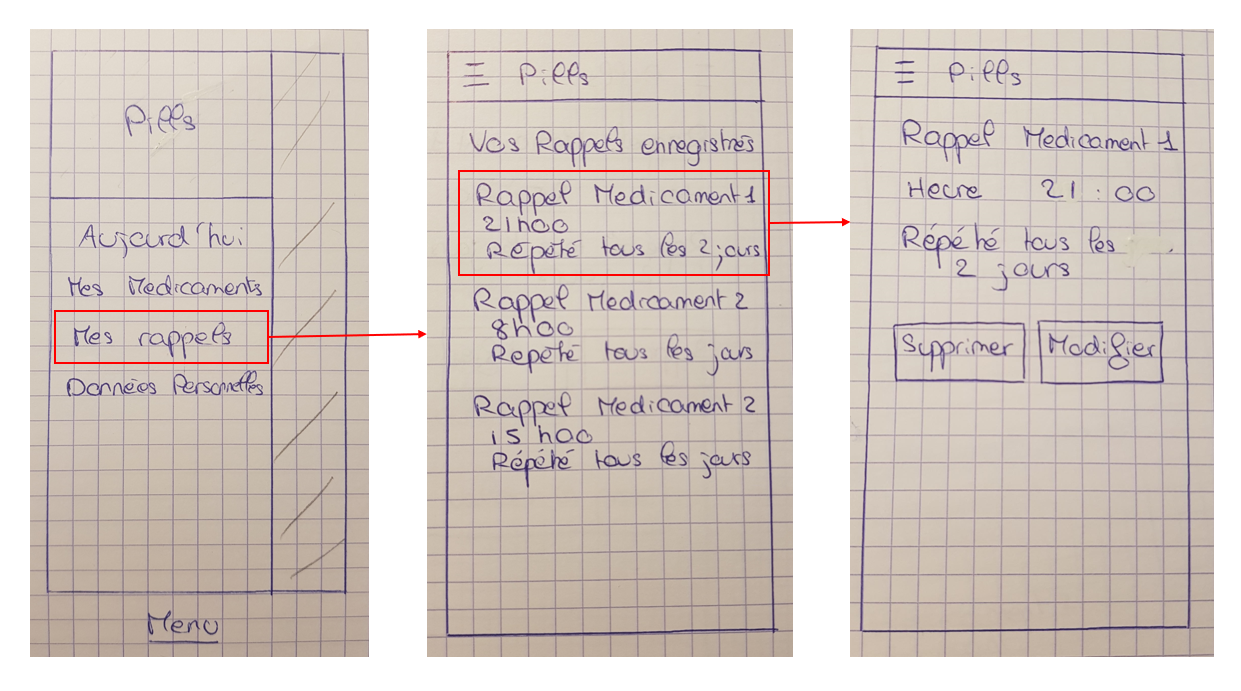


Figure : Maquettes de la liste des rappels

### Données Personnelles

La page “Données Personnelles” propose à l’utilisateur deux fonctionnalités en lien avec ses données enregistrées. Il peut premièrement consulter l’ensemble des données sauvegardées dans l’application. Il sera envoyé sur une page où il pourra consulter l’intégralité des informations stockées dans les tables Medicament et Rappel. La deuxième action que l’utilisateur peut effectuer est de supprimer l’intégralité des données.

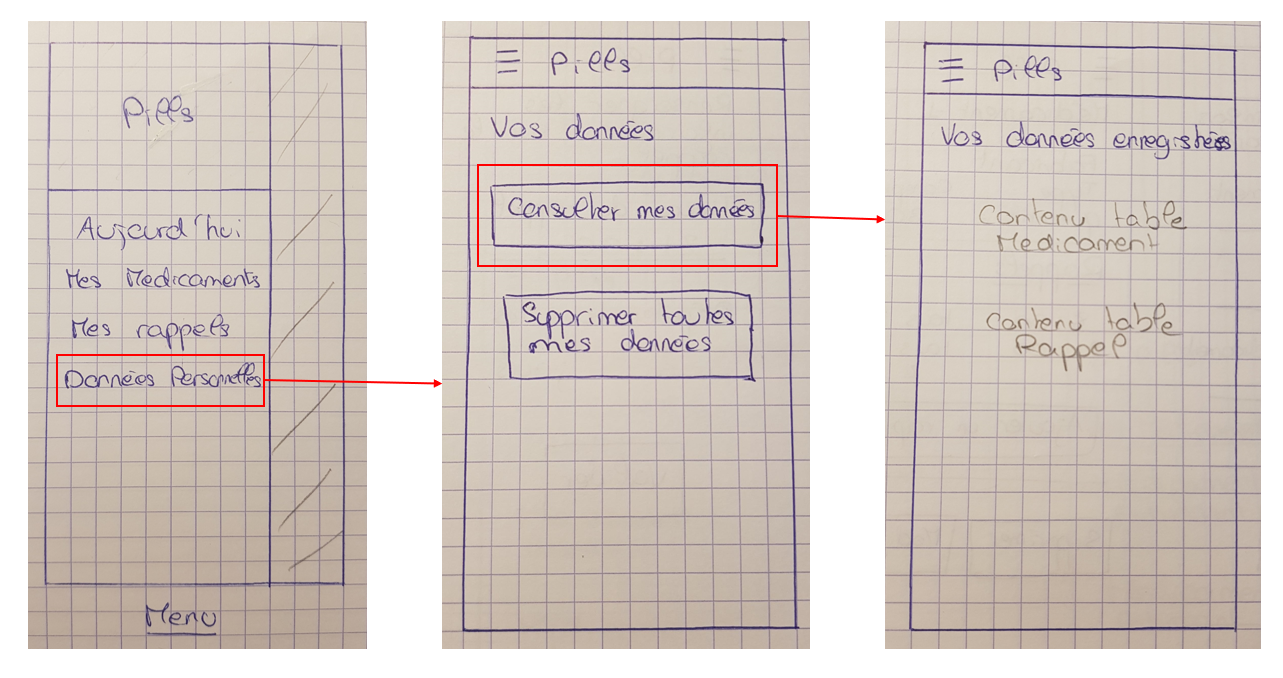


Figure : Maquettes de la page Mes données, et leur consultation

## Conception de l’application Android

En conception Android, l’ensemble des pages de notre application décrites ci-dessus sont des activités. Chaque activité est liée à un layout permettant d’organiser l’interface graphique. Certaines activités, comme l’ajout et la modification d’un médicament ou d’un rappel, possèdent le même layout puisque l’interface voulue est similaire. La navigation d’une activité à une autre se fait par l’utilisation d’Intent, nous donnant la possibilité d’accéder à une nouvelle activité, mais également de passer des paramètres si besoin.

Pour commencer la conception, nous avons dû créer les classes *Medicament* et *Rappel*, qui permettent de définir les objets de type médicament et rappel, qui sont donc la base de notre application et utilisés tout le long de la conception. Ces classes permettent donc de définir les attributs de nos objets, qui sont les mêmes que ceux des tables correspondantes de la base de données. Les classes Medicament et Rappel contiennent également les méthodes de getter et setter nous permettant d’accéder aux attributs des objets ou de les modifier.

Ensuite, pour gérer la persistance de notre application, et les actions sur la base de données, nous avons une classe *MedicamentPersistance*. Cette classe permet donc la définition des tables Medicament et Rappel, ainsi que les instructions permettant leur création, initialisation, modification, suppression ainsi que les instructions pour les vider. Cette classe contient également l’ensemble des méthodes permettant d’ajouter, de modifier, de supprimer, de récupérer un médicament/un rappel précis. Mais également, on trouve les instructions pour supprimer et récupérer tous les rappels/médicaments.

Par ailleurs, pour concevoir notre application, nous avons eu besoin de créer des listes d’affichage dynamique. Pour cela, en Android, pour chaque liste différente, nous avons créé une classe adaptateur correspondante, ainsi qu’un layout pour les éléments de la liste. Les classes adaptateurs nous permettent de récupérer pour chaque élément de la liste les informations correspondantes, et de les afficher selon le layout défini.

Enfin, pour la création du menu en Android, nous avons utilisé un DrawerLayout pour l’ensemble des activités. Ce DrawerLayout se compose d’une toolbar, d’une NavigationView et d’un layout quelconque pour le contenu de l’activité. La toolbar nous permet d’avoir une barre en haut de l’application avec l’icône du menu burger qui ouvre la NavigationView lorsque l’on clique dessus. Cette dernière est constituée d’un en-tête qui possède son propre layout, et d’un menu. Le menu est un layout particulier (de type menu) contenant les divers items du menu. Au niveau du code de l’activité, en fonction de l’identifiant de l’item du menu sélectionné, un Intent se lancera et dirigera l'utilisateur sur l’activité correspondante.

## Conception de l’application IOS

Chaque page de l’application correspond à une story du StoryBoard. Chaque story a une classe viewController associée pour gérer les différents éléments de la page.

Certaines pages de l’application sont composées de TableView pour l’affichage des différentes listes. Le TableView est alors associé à une classe TableVIewCell. Dans les cas où la story contient un TableView, la classe ViewController de la page devra implémenter les protocoles UITableViewDelegate et UITableViewDataSource pour pouvoir gérer l’affichage des éléments d’une liste.

Pour construire la base de données de notre application, nous avons dû créer différentes classes. Pour commencer, nous avons une classe Medicine, qui correspond à la classe médicament et la classe Rappel. Pour gérer la persistance nous avons les classes MedicinePersistance et RappelPersistance. C’est dans ces deux classes que toutes les fonctions liées à la base de données sont écrites : la création de la table, la fonction d’insertion, de mise à jour d’un élément, de sélection avec ou sans paramètres.

La navigation entre les différentes pages de l’application nécessite le passage de différents paramètres. Pour transmettre des paramètres d’une page à une autre, nous avons utilisé des Segue.

# Réflexions et démarches de sécurisation

## Intégration des directives RGPD dans notre application

Depuis le 25 mai 2018, le RGPD est en vigueur, il remplace une ancienne réglementation de 1995. Les organismes traitants ou faisant circuler des données à caractères personnelles doivent s’y conformer. Ce texte vise à renforcer la responsabilité des entreprises. En effet, elles doivent garantir la protection des données personnelles des utilisateurs et pouvoir apporter une preuve de leur conformité au RGPD, à n’importe quel moment à leurs utilisateurs.

Le RGPD est un long texte, mais on peut retenir certains éléments pertinents. La gestion des risques est un point important dans ce texte de loi. Cela concerne le fait de savoir si le traitement des données pourrait causer certains risques, et avoir un impact sur la protection de ces dernières. Cet aspect du RGPD pousse les entreprises à prévoir les potentielles failles de sécurité et la compromission que cela entraînerait pour les données en amont et non plus lorsque cela est trop tard.

Les entreprises doivent organiser leur processus internes, pour permettre un niveau élevé de protection des données. Cette organisation de différents processus implique entre autres la prise en compte dès la conception de la protection des données : une collecte des données raisonnables, porter une attention particulière à la durée de conservation des données, garantir la confidentialité et la sécurité des données… Parmi ces processus, on doit également retrouver une méthode permettant de traiter les demandes, les réclamations des utilisateurs concernant par exemple leurs droits d’accès, de rectification, de retrait de consentement. Enfin, les organismes doivent documenter leur conformité pour pouvoir la prouver aux utilisateurs.

Ces différents éléments du RGPD redonnent un peu de contrôle aux utilisateurs sur leurs données personnelles et apportent plus de clarté sur la gestion de leur protection.

Dans notre application, nous ne collectons pas de données à caractère personnelles. En effet, avec seulement les informations sauvegardées par l’application, nous ne pouvons pas identifier un individu. Cependant, notre application enregistre des informations médicales qui peuvent être sensibles ou très personnelles. Nous avons donc voulu que l’utilisateur puisse avoir un contrôle sur les informations enregistrées par l’application.

Ainsi, une page de l’application est dédiée aux données personnelles de l’utilisateur. Depuis cette page, il peut avoir la liste complète de tous les médicaments et rappels qu’il a enregistrés. Il peut aussi décider de supprimer toutes les données enregistrées d’un seul coup.

## La sécurité des applications mobiles

La mobilité des smartphones et l’utilisation d’applications mobiles peuvent entraîner des problèmes de sécurité au travers des connexions créées. En effet, l’utilisation des smartphones se fait très souvent hors de système privé et quasiment tout le temps sans fil, ce qui peut créer des vulnérabilités.

Les utilisateurs ont très souvent une utilisation et un comportement à risque. Ces menaces générées par les utilisateurs se traduisent par des connexions sur des réseaux ouverts non sécurisés ou encore les actions effectuées sur un réseau inconnu. Or, aujourd’hui presque toutes les applications téléchargées sur un smartphone d’utilisateur lambda présentent une faille de sécurité sous différentes formes. Certaines applications disposent d’autorisations d’accès ou d’utilisation de certaines informations qui ne lui sont d’aucune utilité pour son fonctionnement mais qui menacent la protection et l’intégrité des données de l’utilisateur.

Ces problèmes de sécurité récurrents montrent que les applications ne se préoccupent pas assez de la protection des données personnelles des utilisateurs. Avec le RGPD, les applications vont devoir se soumettre aux nouvelles réglementations sous peine de sanctions. Ces nouvelles mesures seront peut-être un moyen de plus responsabiliser les concepteurs d’application sur la protection des données et l’utilisation des données personnelles. Cela permettra aussi à l’utilisateur d’être plus averti de la façon dont les applications agiront pour garantir la sécurité de leurs données.

# Conclusion